ENEE 640 VLSI Design
Global Placement
Implementation of Simulated Annealing based Placement Algorithm (Timber wolf Algorithm)

Akshitha Chanda

Abstract
Cell placement is an essential step in VLSI physical design automation - it is the portion of design flow that assigns exact locations for various circuit components within the chip's core area. This Report presents the implementation of a Simulated Annealing-based placement algorithm (TimberWolf algorithm) in C along with the use of partitioning tool hMetis, discussing the performance of code and its resulting output.

Introduction

Placement is the process of arranging the circuit components on a layout surface. During the placement steps, the VLSI circuit is seen as a set of rectangular blocks interconnected by signal nets. Placement consists of placing these blocks on a two dimensional surface such that no two blocks overlap, while optimizing the area of surface, the wire length interconnection length between the blocks and the VLSI circuit performance. [2]

Given a placement of cells or modules or blocks with ports (inputs, outputs, power and ground pins) on the boundaries, the dimensions of these cells (height, width, etc.) and a collection of nets (which are a set of ports wired together), the process of placement consists of finding suitable physical locations for each cell on the entire layout [1]. The major constraints of the global placement are that all the cells need to be placed in fixed rectangular space with minimum wire length and minimum overlap.

The wire length measurement function is implemented using HPWL technique and a combination of wire length and overlap is used as the cost function for this implementation.

Algorithm Formulation

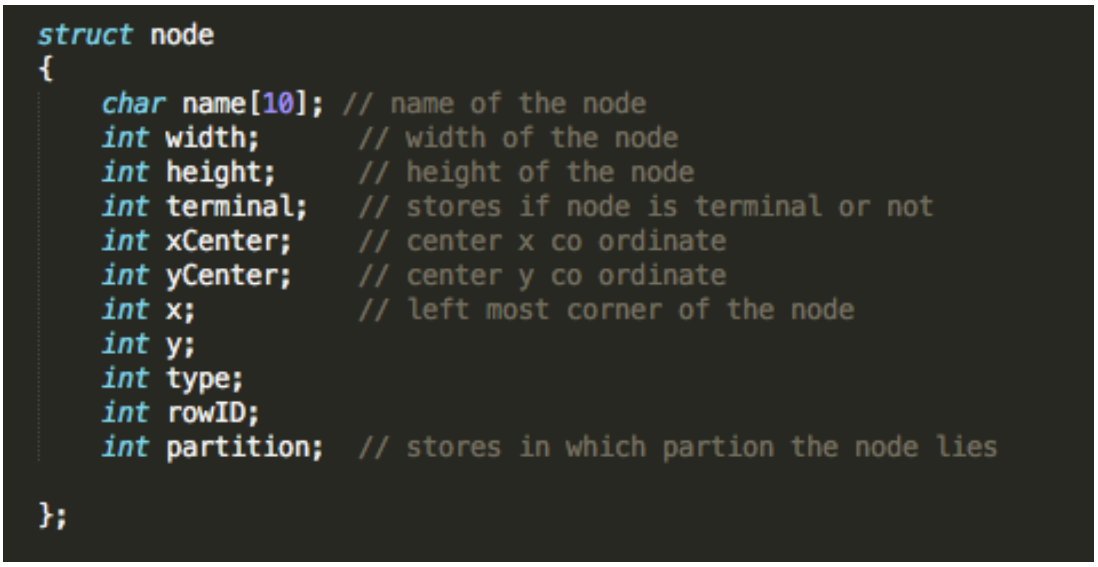
The main goal of the global placer is to find fixed location for each node in area restricted by the terminals with minimized overlap and wirelength. So, based on this costfunction has been formulated as

$$\textbf{\textit{Cost function}} = \alpha * \textit{wirelength} + \beta * \textit{Overlap}$$

Alpha and Beta are the weights of wire length and Overlap in Cost function. Overlap is the total overlap area calculate across the entire chip and wire length is the total HPWL calculated across each net.

Data Structure

At the start of the project I used a B tree data structure to represent the floorplan. The left and right child of each node represent the blocks adjacent to each other. The Btree data structure floorplan was working well with benchmarks of sizes 10 – 300 blocks but I was not able scale it to ISPD02 benchmarks. So, I implemented a struct node which has various attributes like co ordinate, centre co ordinates, width, height of the node and which partition the node belongs to.



```
struct node
{
    char name[10]; // name of the node
    int width;      // width of the node
    int height;     // height of the node
    int terminal;   // stores if node is terminal or not
    int xCenter;    // center x co ordinate
    int yCenter;    // center y co ordinate
    int x;          // left most corner of the node
    int y;
    int type;
    int rowID;
    int partition;  // stores in which partion the node lies

};
```

Fig 1: Data Structure for Node

Simulated Annealing

Global placement is a NP Hard problem and Simulated Annealing Algorithm (Timber Wolf) is used to produce a good solution with a reasonable runtime. Simulate Annealing belongs to the class of non deterministic algorithms. The Algorithm starts with an initial random placement; it moves from the current state to neighbouring state after some moves (small change made to the previous state). The cost function of new state is calculated and the algorithm moves to the new state if the late has a better cost function. The cost curve is convex and therefore it has multiple local minima. Thus, the algorithm should not accept an inferior solution at the local minima, but should apply 'hill-climbing' methods so that it can climb out of the local minima to search for the global minima.

The algorithm uses Metropolis procedure which simulates the annealing process at a given temperature , it receives an input which improves through local search. The main annealing process invokes Metropolis at various decreasing temperatures. An initial temperature T is reduced constantly through geometric progression with a value of **alpha.** The inputs of the Metropolis procedure are the temperature and M value. The difference of oldcost and newcost is calculated. If the difference is less than zero the new placement is accepted if it is greater than zero then a random number is generated and checked against (e- Δh/T) , Δh is the difference between the costs and T is the temperature and it is decided whether to take the hill climbing option by comparing the parameter with a random number. This is done until Metropolis parameter reduces to 0.

The Timber Wolf Algorithm for placement is the modified form of Simulated Annealing. Initially a function **InitialPlacement()** places all the nodes randomly with help of randomly generated functions. Before this the Placement is divided into parts using HMetis software and partition file is used in placement by dividing the chip area into grids and placing the Nodes to particular grids.

The initial Placement is sent to the Metropolis procedure where another random generator is used to select a particular move.

Perturbation

Two perturbation moves are implemented namely Swap and NodeMove.

1. Swap() : Two randomly selected nodes are swapped.
2. NodeMove() : A randomly selected number, x co ordinate and y ordinate are chosen and given node added with random number is moved to that location.

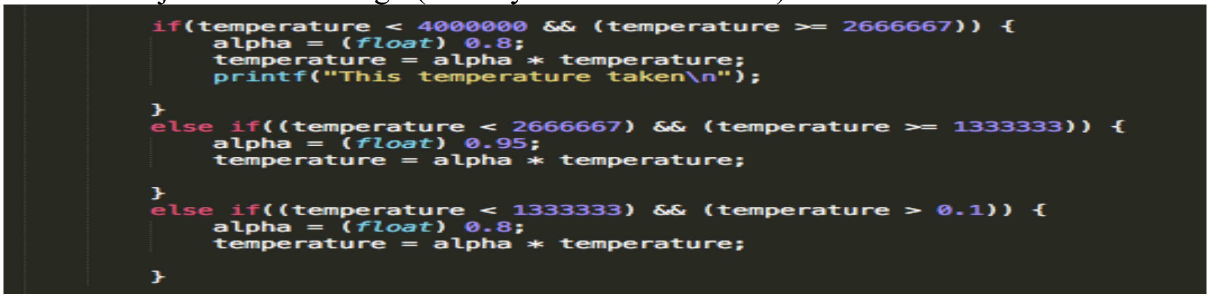
The probabilities of Swap and Node move are 50%

This perturbations are performed until the terminating condition is satisfied.

Temperature Change and Terminal Condition

The initial temperature is 4000000k and the terminating temperature is 0.1k. Depending in which intervals the temperature lies appropriate alpha value is set.

When the rejection rate is larger(usually at the start and end) a conv rate of 0.8 is chosen.



```
if(temperature < 4000000 && (temperature >= 2666667)) {
    alpha = (float) 0.8;
    temperature = alpha * temperature;
    printf("This temperature taken\n");

}
else if((temperature < 2666667) && (temperature >= 1333333)) {
    alpha = (float) 0.95;
    temperature = alpha * temperature;

}
else if((temperature < 1333333) && (temperature > 0.1)) {
    alpha = (float) 0.8;
    temperature = alpha * temperature;

}
```

Fig 2: Temperature Scheduling

Final Placement

After the TimberWolfMetropolis procedure is done a Final placement function is called where all the Nodes are outputted into a .txt file with their respective xCenter and yCenter are printed into file. An additional is written such that the co-ordinates do not cross the border.

Implementation and Results

The global placement algorithm has been implemented in C. ISPD02 benchmarks have been used to test the functionality.

A struct construct is used to to represent Node , 2D array is used to represent nets and placement data is stored in the x and y co-ordinates of the Node.

For parsing the input files (.nodes, .pl, .nets) individual parsing functions have been written and set parameters functions for each attribute is written. A separate function loads the partition value of each node.

Initial placement is done in Initialplacement() function where the cells are set randomly in the fixed boundary. A check boundary function is used to update the perimeter of the chip.Wire length and Overlap function have been implemented separately and are used as part of the cost function.Timber Wolf Simulated Annealing algorithm with NodeMove and Swap functions have been implemented.

The global placer is run on first five ISPD02 benchmarks. To reduce the simulation time for the runs M and Initial temperature have been changed.

BenchMarks	OL = 25%	RunTime	Initial T	Final T	Conv Rate
Ibm01	18957778 (15%)	1 ½ hr	1000000k	0.1k	0.95
Ibm02	6149330 (31%)	~2-3 hrs	1000000k	0.1k	0.95
Ibm03	85260772 (24%)	5-6 hrs	10000k	0.1k	0.95

Discussion

I first started with a B tree data structure combined with simulated annealing where the perturb moves were swapping nodes, delete and insert node. This was working fine with smaller benchmarks but I was not able to scale it to bencharmarks with greater than 10000 nodes. A considerable time has been wasted on this. Then I changed the data structure and implemented TimberWolf to decrease the runtime of the algorithm. I figured out how to use hMetis and used the partition files to place the nodes in fixed grid so that it passes the area constraint. If given more time I would have implemented a min-cut placement algorithm with simulated annealing to do final round of optimization.

To reduce the overlap in random initial placement, I have tried placing the cells row wise. Initially chip is divided into set of row. Macro placement is done(blocks with height > 16) and then cell placement. But this was really complicated as I was not able to fix the max_x pin crossing issue and was not able to scale well as the macro block number increases. But I got good results for ibm01.If that issue was having been resolved, then Row based placement is a better technique than random placement as overlap can be considerably decreased. If given more time I would have done min cut placement (row based) in last step to get better results and reduce the running time.

I observed the overlap percentage could be optimized by setting the weights in cost function. As my algorithm is ridiculously slow I was not able perform exhaustive testing to reduce the overlap to 5 %. Due to shortage of time some simulations are run for lesser parameter and hence not so good solutions obtained.

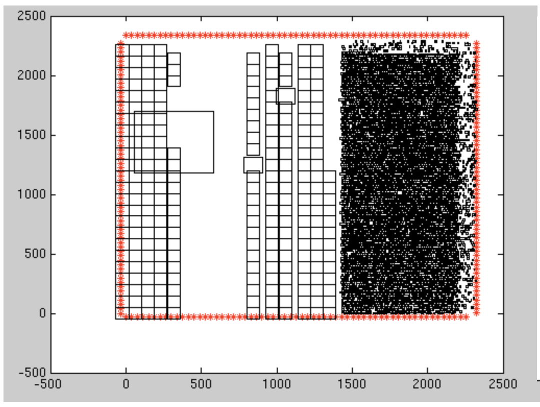
Simulated Annealing Parameter Settings
In general, longer runtime produces better solutions. But setting the initial temperature 4000000k and M parameter to 120 for IBM03 benchmark took more than 5 hours.
So, the next settings of 1000000k and M=50 have been set.

Analysis

1. Increase in the Initial temperature the SA with a high init. temp starts with higher probability of accepting bad solutions, which behaves like random path takingfrom one solution to another and thus effectively avoiding being stuck at local minimal. However, my observation based on repeated simulations is that the increase of initial Temp will increase the runtime but does not always guarantee a better solution.
2. Reducing the terminal Temp: the timberwolf will run a bit longer but it will stop when the rejection rate is larger than the conv rate.
3. Increase the multiplier M(Metropolis factor): better solution will be found at a higher probability since we search more times at each temperature. However, it requires a longer run time (M times run time in worst case).

Plots

IBM01



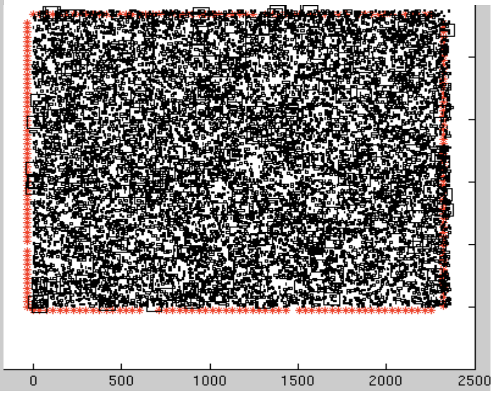


Fig3: ibm01 when Row based placed ibm01 timberwolf + random placement (OL~29%)

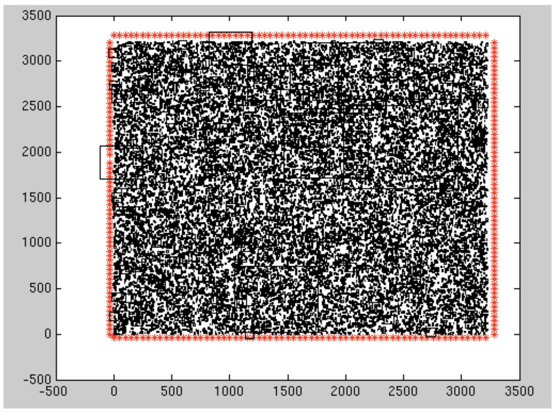


Fig 4: ibm02

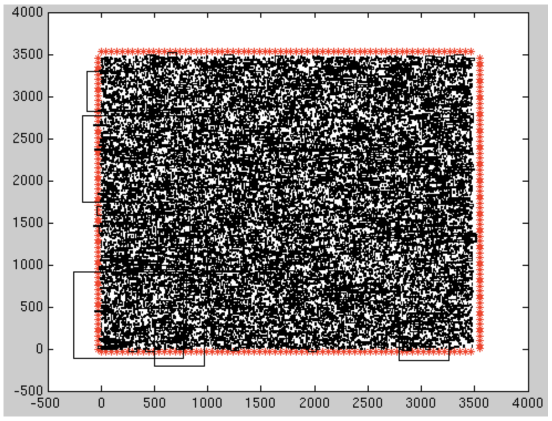


Fig 5: ibm03

References

1. Guo, Pei-Ning, Chung-Kuan Cheng, and Takeshi Yoshimura. "An O-tree representation of nonslicing floorplan and its applications." Proceedings of the 36th annual ACM/IEEE Design AutomationConference. ACM, 1999.

2. The TimberWolf Placement and Routing Package" by Carl Sechen and Alberto Sangiovanni-Vincentelli, IEEE-JOURNAL OF SOLIDSTATECIRCUITS, VOL. SC-20, NO. 2, APRIL 1985

3. "Optimization by simulated annealing" by S. Kirkpatrick, C. Gelatt and M. Vecchi, IBM Computer Science/Engineering Technology WatsonRes. Center, Yofktown Heights, NY, Tech. Rep,, 1982.

4. Floorplanning Tung-Chieh Chen National Taiwan University, Taipei, Taiwan.